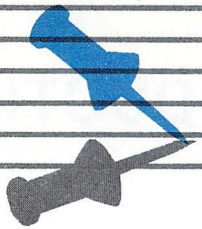
RE RUN

Spring Edition 1985

Great new software programs for the C-64 and VIC-20!



DIRECTORY

C-64	
CALCAID64	1
QUATRO	8
SCREEN	10
SMOKING	13
TAG	18
MEMO	19
JACKETS	21
PLENTY	23
RECORDS	26
SCRAMBLER	34

VIC-20	
SCREEN	10
JACKETS	21
SCRAMBLER	34
CANYON	36
CANYONMOD	36
COLOR	40
BOMBS	50
BOMBSMOD	50
TRAPPED	52
ZXYLON	59
ZXYLONMOD	59
MAZE	61

NOTE: Since some of these programs include features that could overwrite or destroy other programs on the same tape or disk, it is a good idea to make copies of all the programs and use the original ReRUN tape or disk as a backup. ALWAYS read the article in the booklet for operating instructions, memory requirements or special loading instructions before running any program. Also note that some programs are not included on the tape version. Check article in booklet for complete details.

INTRODUCTION

1985 is well under way, and *RUN* magazine is doing better than ever. In 1984, the first two ReRUNs we tried were very well received, but in doing only two a year, we had a rough time deciding which programs to include and which to leave out. Many people wrote to us saying they enjoyed ReRUN Volumes I and II a great deal, but that there were programs they thought we shouldn't have omitted. The solution to this problem is a simple one—more ReRUNs! Twice as many! And more on top of that!

Beginning with this first ReRUN of 1985, we will be coming out quarterly; four ReRUNs a year filled with programs from the best Commodore computer magazine ever published, plus extra, unpublished programs to give you even more for your money! For example, CalcAid 64, a full-blown spreadsheet program, has everything you need for calculating expenses or keeping financial records. It uses 780 cells (26 rows by 30 columns), allows forced recalculation at any time, and has save to tape or disk and many other features. To buy a similar program, you would probably have to pay $50 to $80, or more; yet, for the price of ReRUN, it's yours!

Of course, we also have plenty of programs straight from the pages of *RUN*—programs to help you get more use from your computer. Plenty of K (C-64) is a computer sampler of graphics ideas. Smoking Joe (C-64) takes you from stitchery to sprites, with six tricks to try in your own programs. Print Your Screen (C-64 and VIC) is a utility program that lets you put whatever is on your screen onto paper. Play Me a Color (VIC) is an amusing and instructional use of sound and color.

If you ever wondered

about programming your own maze games, Trapped in the Maze (VIC) not only helps you create your own programs, but is a fine game in itself. Quatro (C-64) and Scrambler (C-64 and VIC) are two more games for those computerists who enjoy an intellectual challenge, while Fly the Grand Canyon (VIC) is a flight-simulation game for the fast-handed gamester. Perhaps one of the oldest children's games ever played makes a computer comeback in You're It! (C-64)—two-player tag without losing your breath.

Since it is almost that time of year again, we have included Tax Records 64 (C-64) to help you get ready to greet Uncle Sam come April 15th. And so that you don't miss those important dates or meetings, Don't Forget! (C-64) is a complete appointment calendar program.

These programs appeared in *RUN* (December 1984 through February 1985), but there is more: four all-new bonus programs never published before! Passage to Zxylon and Bombs Away are two more VIC-20 games to keep you hopping; Bug-in-a-Maze (VIC) is a game that teaches youngsters how to add, and, finally, Making Jackets (C-64 and VIC) lets you make your own disk jackets with your printer, scissors and tape.

This is just the start for ReRUN 1985. We have a lot more in store: more programs from *RUN*, more bonus programs (even a few super-special bonus ReRUNs!), more games, utilities, educational and home business programs—more for your Commodore.

Guy Wright
Technical Manager

HOW TO LOAD

DISK:
To load any of the programs, type:

LOAD " program-name" ,8

then press the RETURN key.
The disk drive should "whirr" while the screen prints SEARCHING FOR (program name). The screen should then print LOADING and then finally READY, with the flashing cursor beneath. Type RUN and press the RETURN key. The program will then begin.

CASSETTE:
Insert the cassette tape into the Datassette recorder, with the proper side facing up. Make sure that the tape is rewound all the way to the beginning. Type:

LOAD "program-name"

then press the RETURN key. The screen will display PRESS PLAY ON TAPE. You should then push the play button on your Datassette recorder.

When the program has been found, the screen will display FOUND (program name). On some Commodore computers, you may then have to press the C = (Commodore symbol) key to load the program. On other Commodore machines, the program will load automatically. Check your owner's manual for specific loading procedures.
When the program has finished loading, you will see the READY prompt and the flashing cursor beneath. Type RUN and press the RETURN key to start the program.

NOTES:
You should use the entire program name as listed to avoid loading programs that have similar titles.
Make sure that if you are loading VIC-20 programs you have the correct memory expansion cartridge (or no cartridge if that is required) plugged in before loading the program.
Some programs are divided into two sections, the main section (the one you should load first) and the MODULE section, which is either automatically loaded when the first section is run or is loaded manually after the first section is run.
Some programs may use special loading instructions, so ALWAYS refer to the article in the booklet for operating instructions, memory requirements, etc.

CalcAid64

BY TRENT BUSCH

This electronic spreadsheet program is one of the most useful tools you'll ever own. You can use it to do anything from balancing a checkbook to performing complex analysis.

This article will take you step by step through the features of CalcAid 64 and give you a sample spreadsheet to try for yourself.

First make a backup copy of the program on disk or tape, then run it and examine the display.

C-64
Printer optional
Disk only

The flashing cursor at the top left of the screen represents the data entry line. Below that is a solid line running across the screen. This is a comment line that CalcAid uses to display important messages and information. The numbers 0, 1 and 2 represent columns. The letters A through T are the rows.

If you don't like the screen colors, you can change them at any time. The f7 key changes the background color, and the f8 key steps through the border colors. To change the text color, simultaneously press the CTRL key with any number key from 1 to 8. Upon your next operation, the entire text will change color.

CalcAid has 30 columns and 26 rows. Each column can display up to nine characters. Notice that only three columns are displayed on the screen. All 30 are there; you just cannot see them all at once. Imagine that you are looking through a window and can only see a portion of the overall picture.

The cursor keys allow you to move this window around

the spreadsheet. Press the cursor-down key, and the spreadsheet will be quickly redrawn with rows B through U. Notice that the text is now the color that you chose. Experiment with the cursor keys until you can place the viewing window over all the columns and rows. Pressing the home key will return the window to A0.

Entering Information

The intersection of a column and a row is called a cell. There are 780 cells that you can use, A0–Z29. There are three types of information that you can enter into a cell: text, numeric or formula.

In order to enter information into a cell, you need to follow a specific procedure. Type in the cell location, row first and column second, without putting in any spaces. Next, type a colon. This separates the cell location from the data. Now you can type in text or numeric data up to nine characters.

```
A0:BUDGET 84
C12:250
```

Text information can contain almost any character on the keyboard, but must not begin with a number or a plus or minus sign. Numeric information, however, *must* start with a number or a plus or minus sign.

After typing in your information, press the return key. If everything was typed in correctly, you should see the data in the proper cell. If you didn't enter your information properly, CalcAid will display a format error message on the display line. To rectify this, simply retype the entire line correctly. Text data will be left-justified, while numeric data will be right-justified.

To replace data, just retype the cell coordinates, a colon and the new data. To clear text or numeric data from a cell, simply type the cell coordinates followed by a colon and then press the return key. This procedure will not clear a formula, however.

Pressing SHFT CLR will clear the entire spreadsheet. For safety reasons, this is a two-step process. First, press SHFT CLR and then answer the question on the comment line. Press Y to clear the spreadsheet. Press N to exit the Clear mode.

Calculations

While you now know how to create neat columns and rows, the real power of CalcAid lies in its ability to do mathematical computations using the data in each cell. For example, you can add cell A0 to cell A1 and put the answer in cell A2. This is accomplished by putting the formula A0 + A1 into cell A2. Here is the

proper format:

A2:{f1}A0 + A1

The f1 key will result in a reverse-character F on the screen. This key is used to access the special features of CalcAid. If you forget to press f1 when entering a formula, the formula will be entered as text and displayed in the cell. Only the result of the computation, not the formula itself, should be displayed in a cell.

A special command allows you to view the formula in a particular cell:

A2:{f1}V

If a formula resides in cell A2, it will be printed on the comment line. The full value of the numeric data in cell A2 also will be printed. This is important, because each column is limited to nine characters. CalcAid will fill the cell with asterisks if the numeric data is longer than nine characters. You will then need to use the View command to examine that cell.

Here are the formulas that this spreadsheet can use for computation:

addition—cell + cell or
  cell + constant
subtraction—cell – cell or
  cell – constant
multiplication—cell*cell or
  cell*constant
division—cell/cell or
  cell/constant
exponentiation—cell↑cell
  or cell↑constant

CalcAid cannot handle complex formulas. A more involved computation can be done by storing the intermediate answer in a spare cell. Extra characters after the second cell or constant will be ignored or show up as a Format error. When typing in a formula, leave out all spaces and be sure to enter the cell first and the constant second.

After you enter a formula and press the return key, you must press the left-arrow key. Wait for the calculation to be performed. During calculations, there will be a working message on the comment line. Calculations are done column by column from top to bottom. Column 1 will be completely done before column 2. This is an important point.

For example, let cell A0 = F9*G6. If cell F9 has a formula in it, the resulting answer will be figured after cell A0 is computed. To overcome this, you should press the left-arrow key twice. After all computations are complete, the spreadsheet will be redrawn with the results displayed in the proper cells. Attempts to divide by 0 will be noted in that cell, as will an overflow note if

an exponentiation calculation is too large.

Commands and Special Features

CalcAid has several other commands that are very useful. The following examples show the proper format for the commands. You may use any cells that you wish. The range must be in a straight row or column, with the first coordinate smaller than the second.

A1:{f1}SUMA2 - Z2

This command puts the sum of cells A2-Z2 into cell A1. Text data is ignored.

Z29:{f1}AVGB3 - B12

This command calculates the average of cells B3-B12 and puts the answer into cell Z29. Text data is ignored.

C12:{f1}MIND0 - G0

This command looks for the minimum figure over a range of cells and puts the answer in cell C12. Text data is ignored.

F5:{f1}MAXZ0 - Z29

This is similar to the MIN command except it returns the maximum value in a range. Again, text data is ignored.

Remember, you can use any cells that you wish, but they must be in a straight column or row.

Z29:{f1}SUMA0 - D29

This formula will not work because cells A0-D29 are in a diagonal.

Here are the rest of CalcAid's special features:

A0:{f1}T

This command makes row A and column 0 titles that are always displayed on the screen. This is helpful in remembering what each cell is supposed to be. As you move your display window around, you can always have a reference to numeric displays. You must always use cell A0 in this command.

A0:{f1}O

This command turns off the Title mode. The cell must always be A0.

C15:{f1}C

This command will clear an individual cell, including the formula, text and numeric data.

F25:{f1}J

This command jumps the display to a particular area of the spreadsheet. Sometimes this is faster than using the cursor keys to move the display window.

D3:{f1}COPD4 - D29

This command is used when you are entering lots of identical information. In this example, the contents of cell D3 will be copied into cells D4-

	0	1	2	3	4	5	6
A	BUDGET 84	RENT	CAR LOAN	GAS	ELECTRIC	TELEPHONE	CABLE TV
B							
C	JANUARY	560	175	110	40	30	15
D	FEBRUARY	560	175	120	40	45	15
E	MARCH	560	175	80	40	30	15
F	APRIL	560	175	70	40	30	15
G	MAY	560	175	30	40	30	15
H	JUNE	560	175	25	60	30	15
I	JULY	560	175	20	75	30	15
J	AUGUST	560	175	20	80	30	15
K	SEPTEMBER	560	175	35	45	40	15
L	OCTOBER	560	175	40	40	30	15
M	NOVEMBER	560	175	55	40	30	15
N	DECEMBER	560	175	80	40	40	15
O							
P	---	---	---	---	---	---	---
Q	TOTAL	6720	2100	685	580	395	180
R							
S	MINIMUM	560	175	20	40	30	15
T							
U	MAXIMUM	560	175	120	80	45	15
V							
W							
X	---	---	---	---	---	---	---
Y	BUDGET	560	175	57	48	33	15
Z	---	---	---	---	---	---	---

	7	8	9	10	11	12	13
A	GASOLINE	FOOD	CHARGES	CLOTHES	INSURANCE	MISC.	TOTALS
B							
C	50	450	100	75	75	250	1930
D	60	450	100	75	75	250	1965
E	60	450	100	75	75	250	1910
F	60	450	100	75	75	250	1900
G	65	475	100	75	75	250	1890
H	80	475	100	75	75	250	1920
I	90	475	100	75	75	250	1940
J	75	500	100	75	75	250	1955
K	70	500	100	75	75	250	1940
L	70	500	100	75	75	250	1930
M	80	525	100	75	75	250	1980
N	100	525	100	75	75	250	2035
O							
P	---	---	---	---	---	---	---
Q	860	5775	1200	900	900	3000	23295
R							
S	50	450	100	75	75	250	1840
T							
U	100	525	100	75	75	250	2120
V							
W							
X	---	---	---	---	---	---	---
Y	72	481	100	75	75	250	1941
Z	---	---	---	---	---	---	---

Table 1. *Sample printout of CalcAid64 program.*

```
C 13  SUMC1-C12    D 13  SUMD1-D12    E 13  SUME1-E12    F 13 SUMF1-F12
G 13  SUMG1-G12    H 13  SUMH1-H12    I 13  SUMI1-I12    J 13 SUMJ1-J12
K 13  SUMK1-K12    L 13  SUML1-L12    M 13  SUMM1-M12    N 13 SUMN1-N12
Q 1   SUMC1-N1     Q 2   SUMC2-N2     Q 3   SUMC3-N3     Q 4 SUMC4-N4
Q 5   SUMC5-N5     Q 6   SUMC6-N6     Q 7   SUMC7-N7     Q 8 SUMC8-N8
Q 9   SUMC9-N9     Q 10  SUMC10-N10   Q 11  SUMC11-N11   Q 12 SUMC12-N12
Q 13  SUMQ1-Q12    S 1   MINC1-N1     S 2   MINC2-N2     S 3 MINC3-N3
S 4   MINC4-N4     S 5   MINC5-N5     S 6   MINC6-N6     S 7 MINC7-N7
S 8   MINC8-N8     S 9   MINC9-N9     S 10  MINC10-N10   S 11 MINC11-N11
S 12  MINC12-N12   S 13  SUMS1-S12    U 1   MAXC1-N1     U 2 MAXC2-N2
U 3   MAXC3-N3     U 4   MAXC4-N4     U 5   MAXC5-N5     U 6 MAXC6-N6
U 7   MAXC7-N7     U 8   MAXC8-N8     U 9   MAXC9-N9     U 10 MAXC10-N10
U 11  MAXC11-N11   U 12  MAXC12-N12   U 13  SUMU1-U12    Y 1 AVGC1-N1
Y 2   AVGC2-N2     Y 3   AVGC3-N3     Y 4   AVGC4-N4     Y 5 AVGC5-N5
Y 6   AVGC6-N6     Y 7   AVGC7-N7     Y 8   AVGC8-N8     Y 9 AVGC9-N9
Y 10  AVGC10-N10   Y 11  AVGC11-N11   Y 12  AVGC12-N12   Y 13 SUMY1-Y12
```

Table 2. *Printout of formulas used in demonstration of program in Table 1.*

D29. Only text or numeric information can be copied. Formulas must be typed individually. This works with rows or columns.

Press f2 and you will see a maximum precision display on the comment line. This command does not affect the accuracy of the calculations. It rounds the number only for display purposes. Use the View command to see the full value. Press a number from 0 to 6. Zero means integers and 6 means six decimal places. CalcAid is automatically set up for two decimal places. This command is only for numbers that are computed by a formula. If you want two-place decimals on all the numbers, you must type them that way.

Press f4 and follow the screen directions to save the spreadsheet to tape or disk. Pick out a logical filename under which you can save the spreadsheet.

Press f3 and follow the screen directions to load the spreadsheet from tape or disk.

To print the spreadsheet on paper, press f5 and follow the screen directions. You can print the whole spreadsheet or any portion of it. You will need to know the top-left cell coordinate and the bottom-right cell coordinate of the area that you want printed out. If you specify more than seven columns, CalcAid will automatically break the printout into sections for you.

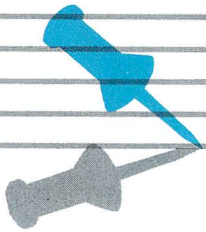
To print the formulas on paper, press the f6 key and follow the screen directions.

The possible uses for this program are innumerable. To start, you might want to copy the budget planner spreadsheet (see Table 1) and adapt it to your own needs. All the for-

mulas are listed in Table 2. The more you use this program, the more applications you will find for it.

If you have any questions or want to discuss the program, just drop me a line along with a self-addressed stamped envelope. R

Address all author correspondence to Trent Busch, 716 Rascoe Ave., Muscatine, IA 52761.



Quatro

BY SOL STEINBERG

This game adds a new dimension to Tic-Tac-Toe. You must get *four* in a row to defeat your opponent.

This strategy game is like Tic-Tac-Toe, but more difficult. The playing field is seven columns wide and eight rows high. You play by choosing a column, and the computer then enters your color in the *lowest* unoccupied square in your chosen column. The first player to occupy four squares in a line (horizontally, vertically or diagonally) wins.

C-64
Disk or cassette

The program will ask if you want instructions. Type Y for yes or N for no. Next, you'll be asked if you want to play against the computer or another player. Type C or P.

If you type P, the game will start and one player will be red and the other black. The computer's only functions in this situation will be to act as a recorder and an umpire; it will neither allow illegal moves nor overlook a win.

If you type C, you'll be asked for the play level. Type A for amateur, P for pro or W for world class. At the pro level, the computer looks ahead to see that it doesn't create new opportunities for you to win. At the world class level, the computer sets traps and waits for you to fall into them. If you are playing against the computer, you will always be red.

Each player will be asked "Which Column?" Simply type the column number you want to play, and the computer will fill the next vacant square

with your color and will sound a tone. It also will check to see if that move gives you a win and if any squares are left to be played. A winning move receives a fanfare. After each game, the program will ask if you want to play again. Type Y or N. The starting player varies with each new game.

If you play against the computer, the computer's first play is random, to provide variety in the games. Try to control the center of the playing field and avoid creating an opportunity for your opponent to win. You must keep in mind that each time you move to a square, the square above that one becomes available to your opponent.

In designing the program, I placed the most frequently used subroutines near the beginning, to obtain maximum speed. Despite that, the computer takes nearly 15 seconds per move; it must consider every possible combination of four contiguous squares that includes the available square in the chosen column.

I hope you will find this game a welcome break from arcade and adventure types. R

Address all author correspondence to Sol Steinberg, Apt. Q9, Hyde Park Apts., Bellmawr, NJ 08031.

Print Your Screen

BY ROBIN FRANZEL

How often have you been working at your computer and wanted a printout of the screen? This article presents a program, written in machine language for the Commodore 64, that will print your screen whenever you press the f1 function key or call the print routine from Basic.

The program is interrupt-driven, which means the keyboard is scanned every 1/60 of a second to see if the f1 key is being pressed. If it is, the screen is then printed. The IRQ vector technique enables this utility to work even while executing a Basic or machine language program. It can also be used while the disk wedge program is in the computer, as long as you load and run the wedge first.

The screen dump utility is also a wedge, and when it does not find the f1 key pressed, it transfers control to the DOS wedge program, so that both utilities can function properly together.

The loader program will ask if you want uppercase or lowercase letters to be printed. It's easy to switch back and forth, even after the program is loaded, just by changing the command sent to the printer. This is done with a Poke, from either your program or Command mode. To change to lowercase letters, enter POKE 49203,7; to change to uppercase, enter POKE 49203,255.

If you're running a program that uses the f1 key, it's very easy to change the print key. Simply type POKE 49184, with the key code for location $C5 (197 decimal). For example, to make the f7 key print the

Run it RIGHT

C-64 and VIC-20
Printer required
Disk or cassette

screen, enter from Command mode: POKE 49184,3.

Pressing the run/stop and restore keys will disable the utility. SYS 49152 will re-enable the screen print key.

Now, what *can't* you do when this utility is in place (if you still want it to work)? First, you cannot use locations $C000 to $C0F9 (that's 49152 to 49401 to you decimal folks). Four bytes are used by the program at $02A7 to $02AA (679 to 682), and zero page bytes $FB to $FE (251 to 254) are also used. That's it!

I hope that you find this program useful and interesting. Have fun with it! ®

Address all author correspondence to Robin Franzel, 5521 Harvey Lane, Alexandria, VA 22312.

The VIC-20 Screen Dump Utility

The VIC-20 version of the screen dump program is completely relocatable, and "hides" itself below the current top of Basic memory. It is, therefore, able to function with the VIC wedge program in memory. It also determines where screen memory is located, making this screen dump program compatible with all memory configurations.

To use this program, load and run the Basic loader. Then, whenever a print is desired, simply press the f1 key.

When the Basic loader program is run, you are asked whether upper- or lowercase print is desired, and the appropriate command is sent to the printer. The start address is provided, so that you may change the Printer command. For example, to change from uppercase to lowercase, enter:

```
POKE (start address) + 54,7
```

If the Print routine is to be used from within a user program, the program should use the following line of code:

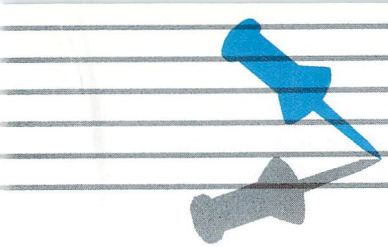
```
100 POKE SA + 245,96:SYS SA + 44:POKE SA + 245,76
```

SA indicates the start address of the program. Of course, it is your responsibility to load the screen dump program and to determine the start address.

The following locations are useful to note:

- SA = start address of utility. If run/stop and restore keys are pressed, the f1 key is disabled. To reenable it, do a SYS SA.
- SA + 33 = key code being checked. Normally, this is a 39 for the f1 key, but this location may be Poked with another code (see the *Programmer's Reference Guide*, p. 179) for use with programs that utilize the f1 key.
- SA + 44 = the Print routine. This is the entry point from user programs.
- SA + 54 = Printer command. Poke to 7 for lowercase, 255 for uppercase.

I hope you will agree that the VIC-20 version of the Commodore screen dump utility is a valuable addition to VIC-20 users' libraries.



Smoking Joe

BY MARK JORDAN

Warning: The Surgeon General has determined that the following six tricks and accompanying program will make it impossible for you not to be more creative in your sprite programming.

Sprites are the greatest things since plastic straws, but their potential is largely untapped. Documentation for them is not hard to come by, but most of it is too technical, depressingly uncreative and fails to demonstrate their unique capabilities. I've discovered six programming tricks that have added many dimensions to my sprite programming. Smoking Joe, the program included with this article, demonstrates all these techniques. If my explanations of any of the tricks confuse you, just run the program, and Joe will clear the smoke.

One last note: Three of the sprites in this program are interrupt-driven by a machine language routine. If you don't know machine language, don't worry about it. You'll still be able to glean plenty from the Basic portion of the program.

C-64
Disk or cassette

Alpha Sprites

Sprites are excellent places to put words because you can push them around very handily and can needlepoint them into any design you want. For example, if you wanted to show a plane in the sky, pull-

ing a message behind it, you could jerk letters eight pixels at a time, using a complicated printing routine, or you could do it professionally with a sprite.

But wait, you say. A sprite is only 24 pixels across. At eight pixels per letter, that's only enough space for three letters. True, but who said you have to use eight pixels per letter? You can halve that amount and still get crisp letters. . .with this trick.

Use the Horizontal Expansion mode. If you've never done it, it's easy. Just turn on the proper bit (bit 0 for sprite 0, etc.) on register 53277 (D01D in hex, V + 29 with V at 53248), and your sprite will double its width. The Horizontal mode will double the thickness of all your vertical lines. That way, to form the letter H, you only need one pixel for each vertical side, another for the crossbar and a fourth as a separating space from the next occurring letter. An L could be made with only three pixels (including separating space). "HELLO" in this mode only requires 16 dots in the X direction—a mere two-thirds of your total 24. You could create words up to eight letters long.

As you know, sprites are 21 pixels deep. Capital-letter alpha sprites need five pixels for most letters and another for a separating space, so you'll probably only get words that are three letters long.

Need some ideas where to use word sprites? How about a cartoon-type balloon above another sprite's head, expressing a monosyllabic emotion such as "Ouch!" Or try one as a warning message scrolling across the screen, like "Duck!" or "Run!" Smoking Joe uses two alpha sprites—"Puff" and "X-Ray." Check them out—you'll be amazed at how charming and useful alpha sprites can be.

Window Sprites

This trick utilizes the background priority register, 53275 ($D01B). The Default mode sets all bits to 0, which causes the sprites to always have priority over background data. Admittedly, this is often what you want, but reversing the situation will give you some clever effects.

Print a message on the screen in the same color as the background, so the mes-

sage can't be seen. What good is this invisible message? A lot, if you use a sprite to uncover it.

Now make a simple solid box sprite like this:

```
FOR T = 12288 TO 12350: POKE
T,255: NEXT
```

Place it under your invisible message. The sprite can be made to move, via joystick, keyboard, loop or whatever, across the hidden letters, and, if bit number 0 on 53275 is set to 1, the letters will show up on top of the sprite. The effect is one of a roving window or periscope.

You can use this technique to hide secret information on the screen; only a sprite can uncover it. You may also use this technique to keep a sprite in safe territory. Individual letters, functioning as clues, can be dispersed across the screen. Again, you are limited only by your imagination. Observe the use of the window sprite in Smoking Joe—a cigarette package. Ah, but what wisdom it unearths.

Layered Sprites

Sprites that are stacked one on top of another are layered. They're easy to keep together as long as you always move them in harmony. The uses of layered sprites are many and varied.

You could design one sprite as a body minus some parts, then use another to supply the missing elements. You could then animate the layered parts independent of the foundation sprite.

For example, you could use a heart-shaped sprite centered on a sprite boy's chest. Then, when sprite boy meets sprite girl, you might start sprite boy's heart pulsating by jumping it in and out of Expanded mode (love at first sprite?). Is such an effect worth an extra 63 bytes of data and five or so program lines? You bet. Just watch Smoking Joe's chest (and cigarette smoke) rise and fall, and you'll see the value of this ploy.

Another use of layered sprites utilizes the collision-detection register. You program one of the layered sprites (but not the other) to detect collisions. Imagine having a sprite creature whose innards are actually a different sprite. Since only the innards detect collisions, a bullet you shoot must hit a vital organ, or no luck.

You might decide to let one of the stacked MOBs (moveable object blocks—the technical term for sprites) function as a window sprite. You could create an eyeless Martian, who must "see" through his hands. Only his hands, which would be a different sprite entirely, would have background priority. He could then "feel" his way across the screen to find invisible information.

Multiple Sprites

It's tough to make a complete body in a 24 x 21 grid of dots. A superficial solution to this problem is to expand the sprites, which sometimes helps, but you lose resolution and still don't have any more realistic figure than you did before—just bigger.

Why not use two sprites to accomplish the same thing? This trick is so obvious you've probably thought of it, but have you actually done it? It really works.

Combined with animation, you can get your sprites to get down and get with it. Make a person in two halves and let the legs walk. Smoking Joe is a prime example of this technique. If you incorporate the reverse sprite feature, you can achieve a comical effect by having Joe's head turning from side to side while his body remains stationary.

Evolving Sprites

By altering the data that constitutes a sprite, you can make it do interesting things. For instance, you can watch it dissolve or fill up. For what? Well, how about this.

You design a milk bottle and then use a data-altering routine to slowly fill in the bottle from the bottom up. If the bottle sits beneath a multiple sprite cow, then you've got some fun going. A thirsty sprite cat could reverse the process. Smoking Joe's lungs aptly describe this technique as the blackness of his disease spreads. Lines 190-200 perform this by changing the data in the sprite block, in reverse order and one at a time, to 255s.

The dissolving idea is a little harder to describe. Smoking Joe shows off this technique as his final puff of smoke dissolves. Lines 260-270 should be studied to see how random numbers (between 0 and 255), used with AND along with the existing data, cause the sprite to

slowly wisp away.

You can do a sprite kaleidoscope by Poking in all the necessary numbers to get a sprite going, then filling the data block like this:

```
100 FOR T = 12288 TO 12350 : R = INT
    (RND (0) * 256) : POKET,R : NEXT
    : GOTO 100
```

Cover Sprites

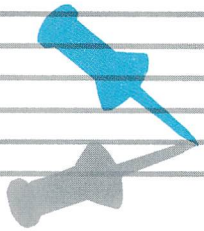
This technique is based on the ability of sprites to move one pixel at a time. By creating a solid block sprite and giving it the same color as your screen's background, you can effectively, dot by dot, either hide or reveal something.

You might have a stick of dynamite with a long fuse. Your cover sprite could slowly but surely make that fuse disappear. Or you could use a timer technique similar to many professional games like Frogger, where a solid bar represents remaining time. The bar could be printed on the screen, and a horizontally expanded solid sprite bar (which would be invisible because it's the same color as the screen background) would be placed next to it and then incremented a pixel at a time until the printed bar was covered.

Smoking Joe utilizes background data as a cover sprite. In Joe's introduction, a reversed space that is the same color as the screen's background was printed over Joe's cigarette. Then, when the program tells you of his bad habit, the space is deleted and there dangles his vice.

Well, there they are—six nifty ways to put new life into your 64's moveable object blocks. Now it's your turn. Get busy and put Smoking Joe to shame. R

Address all author correspondence to Mark Jordan, 70284 C.R. 143, Ligonier, IN 46767.



You're It!

BY GERALD CODDINGTON

How would you like to play tag indoors? Here's a BASIC program that requires two joysticks, so you and a Friend can chase each other around the screen.

Before you load and run Tag, make sure your two joysticks are plugged in. The opening screen allows you the option of reading instructions if you're not familiar with this game.

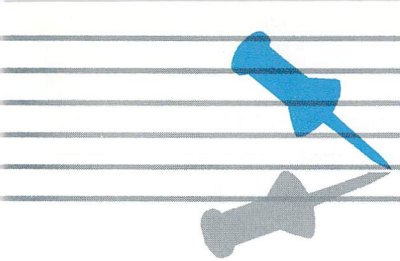
Run it RIGHT

C-64
Two joysticks required
Disk only

The object of this version of Tag is to accumulate two minutes' worth of free time. Your free-time total grows whenever your opponent is It.

Note that there are two hiding spots located at the top and bottom sections of the screen. Use them to your advantage. R

Address all author correspondence to Gerald Coddington, Rt. 3, Box 296, Gilmer, TX 75644.



Don't Forget!

BY RICHARD LOVETT

Here's the perfect program for people with a busy schedule and a bad memory. It lets you create an appointment file and has a built-in calendar as well.

Instant Memo allows you to enter up to three reminder memos for any day of the year, review or change them, and, if you have a printer, make a hard copy of one or more months' worth of entries.

C-64
Printer optional
Disk or cassette

The program can also display an accurate calendar for any month between the years 1700 and 3099. Any day for which you've entered a memo will be highlighted in reverse video on the calendar, and you can print a hard copy of the calendar itself. (Free wall calendars are getting harder and harder to come by.)

Getting Started

When you run Instant Memo, you will first see an introductory menu allowing you to load an existing file from disk or tape, create a new file or simply view a month's calendar. If you choose the calendar option, you will be returned to this menu after viewing the desired month.

Selecting the "new file" option sends you to the main Input routine. Here you specify a day, and then view, enter or delete any desired memos for that date. (Memos can be up to 77 characters long, although

they will display more neatly if kept under 40 characters. Don't use any commas, colons, semicolons or quote marks.) To select a day, you first enter the month as a number between 1 and 12, then the day and year. Use all four digits of the year.

Actually, specifying the year does not prevent a memo entered for, say, July 4, 1984, from also being displayed if you call up July 4 of some other year. Making each day of several years entirely separate would have required dimensioning the arrays in line 10 several times larger, thus quickly exhausting the computer's memory. Putting an "84," "85" or other designation at the end of entries from different years is an easy way to overcome any confusion. Entering the year *does* govern whether or not you can enter or view a memo for Feb. 29, which only occurs in leap years.

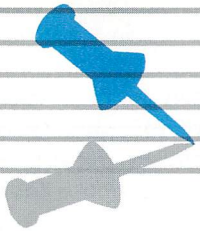
The main menu, accessible from most other modes of the program, includes options to view a month's entries on the screen, view more calendar pages or save the file to tape or disk.

Is Today the 6th?

Lines 385–710 compute and display the calendar. Line 420 reads in the name of the month and its length in days. Then the formula in lines 430–480 determines which day of the week the first of that month falls on. Using that day as the starting point, lines 500–670 Poke the dates on the screen and highlight any days containing memos.

A modified screen dump in lines 940–1005 prints a hard copy of the calendar. A screen dump is a routine that sends the contents of the screen to a printer or other device. Because the upper limit of LI in line 950 is 19, however, the routine will reproduce only the top 19 screen lines. This avoids cluttering up the printout with the prompts at the bottom of the screen. R

Address all author correspondence to Richard Lovett, 6649 Oak St., Kansas City, MO 64113.



Making Jackets

BY GLENN W. ZUCH

Disk Jacket Printer complements Michael Broussard's disk directory utility program ("Calling Disk Drives to Order," *RUN*, April 1984), which provides a very handy and useful printed directory of disk files formatted to fit protective jackets.

Disk jackets normally become useless (or frustrating to use) due to the following conditions:

- Frequent use; the jackets become dog-eared and torn.
- Time and change; the directory labels you pasted on them no longer apply because you scratched and replaced all or most of the files on the disks.
- Disorganization; frequently used disks (especially utility disks) always seem to end up in the wrong jackets.

Run it RIGHT

C-64 and VIC-20
Printer required
Disk or cassette

Disk Jacket Printer is written for most printers. If you have a Commodore printer or one that supports Commodore graphics, you should find it easy to substitute some of the characters in the four strings that compose the jacket outline (lines 30–60).

I used the asterisk to make the label border, but you can use any character (simply change the asterisk in lines 200 and 240). By the way, the C-64 version of this program will also work on Commodore's new Plus/4 computer.

I have used screen color changes for each program prompt. That way, you will

immediately know when something new is about to take place. If you have utility programs that cause you frustrations because you sometimes overlook the prompts, you may wish to add color Pokes as an improvement to them.

The label-printing routine (lines 90–100 and 190–250) can be used in many of your programs to print centered titles or headings. Just modify the 72 in line 220 to correspond to the total width of your printed page. (A 60-column printout would mean replacing the 72 with a 60.) You can use the same routine to center messages or titles on the screen. Change the 72 to 40 and remove the #4 from lines 230–250.

I have found it convenient to print my jackets on different colored papers to facilitate locating my files. You, too, may find this color-coding helpful.

The best way to fold the printed jacket is to first fold along the line that becomes the bottom edge of the jacket, then lay the jacket face down (label down) and fold over the tabs. A small amount of any white glue or similar paper cement is all that is required to complete your jacket.

Here's to a better organized disk-filing system. R

Address all author correspondence to Glenn W. Zuch, 183 Hagen Ave., Tonawanda, NY 14120.

Plenty of K

BY ELIZABETH OMAN

With this article and program, you can create various sizes and shapes of graphics characters, based on the letter and number patterns.

As Janelle enters her first computer class, the computer screen displays HELLO JANELLE in extra large letters. Janelle stares at the screen in disbelief—talk about a friendly computer! Janelle will be an enthusiastic student from day one!

My husband and I teach "Introduction to Personal Computers" at a recreational vehicle (RV) park in the Rio Grande Valley in Texas, and we use this form of greeting on our students. We teach the class in our motor home, so some improvising is necessary.

We do not have a traditional blackboard, and a 9- × 12-inch magnetic memo board doesn't always do the job. One day we entered some Basic terms such as bit, byte, RAM, ROM and K on the monitor in large letters so our students could better see and remember them. By using cross-stitch patterns for the letters, their sizes ranged from 3 × 5 spaces to 22 × 24 spaces.

C-64
Disk or cassette

Try It

A good source of different sized and shaped letters and number patterns is a sampler, which originally was a piece of embroidery.

A sampler was used before the days of printed material and later was often used as a

reference for stitches and patterns. Sometimes, it was hung on the wall so the young children in the household could practice their alphabet and numbers from it. Samplers were popular during the Colonial period in this country, and people still make them today.

You can use these same patterns on your computer. There are hundreds of "stitches" available, plus many patterns for pictures, including some that will fit into the grid for a sprite. There are also magazines and books on cross stitching and needlepoint.

You'll find a few books to get you started in the library's 740s section, if your library uses the Dewey decimal system of classification, or the TT section, if it uses the Library of Congress classifications.

Plenty of K

The program with this article works with the letter K. The program shows 12 different sized Ks, from 3 x 5 spaces to 22 x 24 spaces. Various graphics, letters and colors are used to demonstrate them in the sampler section of the program.

After you view the various sizes and techniques used, you can go on to modify the large K, in as many ways as you can imagine, by changing the graphics and the color of the stitch. Black is used as the background color for the whole program, as more colors show up better on black than on any other color. White is a close second, but on our TV monitor, it created too much of a glare.

When you are experimenting with the large K, do not use a colon, comma, quotation mark, RVS on or RVS off by itself. If you use more than one letter (for example, WW), you'll find that on pressing the return key, the newly created letter will be too large for the screen. Try again with only one character. Later, you can try more involved graphics by using "[RVS on] [any character]".

Using a space after the RVS on will give a nice block effect to an otherwise fancy K. Always remember to use the return key after entering your choice of stitch. If you want to try another, press any key or the space bar. Some incorrect entries will make the screen scroll. By pressing the run/

stop key and then entering RUN, you may get back into the program.

To exit this part of the program, use ZZ for your character, and you'll be returned to the menu, where you may then go on to the third part of the program. Use it to help you design your own sampler, title page, vocabulary features and so on.

First, determine how many lines of text you will have (no more than four). If you were going to do the title (SAMPLERS FOR THE MAKING, for example), you could put each word on a separate line. Going through the exercise, you would enter 4 and be told that the maximum height of the letters would be seven. You would then enter the number for the line with the longest text in it. In this case, SAMPLERS is eight letters long, so you would enter 8. The maximum width for any letter would be six.

The program takes care of putting one space between each letter and row. If you want more than one space between each letter or row, you must refigure by subtracting 1 from the answers given for each additional space, then going to your sampler book of patterns and finding appropriately sized letters.

To exit the whole program, just press the run/stop and restore keys. Otherwise, you'll keep returning to the menu.

This program should have uses in the classroom, for advertising bulletins, for the visually impaired or for titles for a slide show. If you develop any new ideas while you're experimenting, I'd like to hear about them. R

Address all author correspondence to Elizabeth Oman, 800 North Second, Lindsborg, KS 67456.

Tax Records 64

BY GARY V. FIELDS

Load Tax Records 64 into your C-64, and April 15, tax day, will be a lot easier, and perhaps cheaper, to face.

To assist me in using Tax Records 64, I keep one large envelope where I store every bill, check stub, receipt and so on until the first of the month.

I then enter everything into Tax Records 64 and I divide the income and deductions into two other envelopes, which I place in storage. The large envelope is then empty again, ready for another month's receipts.

C-64
Printer optional
Disk only

I make a separate file of each month's records (June 84, for example). Then, at the end of the year, I merge all 12 files into one, named Tax File 84.

In Tax Records 64, there are two main menus: File Options and Work with File (name).

You cannot get past menu 1, File Options, until you create a file and store it in memory. Press N for New File, give the file a name and answer the prompts.

Respond to the Category prompt by pressing either I for income, D for deduction or ! to end the file. The program will allow you to enter only these three characters.

If you choose I, answer the Subcategory prompt by pressing either W for wages, F for farm, D for dividend, O for other, I for interest, R for rental or B for business.

If you choose D (deduction), answer the Subcategory prompt by pressing either I for interest, M for medical, E for education, B for business, T

for tax, C for contribution, R for retirement fund or O for other.

The program has checks to prevent you from entering wages or farm as a deduction, or education, tax or medical as income.

Next, enter the source of the money (at the Source prompt). This must be a minimum of four letters, a maximum of ten letters. The program will not allow you to break this rule. Be mindful that the first four letters will be used later to search for a specified source. So, if you entered interest from three different banks as Bank 1, Bank 2 and Bank 3, it would be better to enter the banks as 1 Bank, 2 Bank and 3 Bank, so you could later search for them separately if need be.

When responding to the Date prompt, always enter the month, followed by the day, and always use five characters (for example, enter 01-02 for January 2).

You may enter anything in response to the Record/Receipt prompt. Your answer must be from one to six characters long (for example, CK#123, or STUB, or NONE).

The Amount prompt is last. Enter dollars and cents. Don't use commas. Don't use a dollar symbol. The program will not allow the first character to be anything other than a number or decimal point. You must enter cents, even if it's only ".00".

Next, you will be asked if all the data is correct. If you select N, the cursor will return to the first item. Press the return key until the cursor rests on the incorrect item. Change the data and press the return key until you are asked if it is correct again. If it is, enter Y and press the return key. When you are finished, enter ! as the category, and the file will be written to disk and closed.

Press the return key, and you'll be returned to menu 1, File Options.

Menu 1's Options

-LOAD EXISTING FILE-

After a file has been created, you call it into memory by selecting L from the File Options menu. You will be prompted with "Recall File." Enter the name of the existing file (? and * wild cards are legal). The file will be printed on the screen as it is recalled, and a total record count will be displayed.

Remember, this program is dimensioned to handle 501 records. Do not exceed this unless you first increase the DIM statement in line 340. My total count for 1983 was 226 records, so 500 records should be adequate for the average taxpayer.

After a file is in memory, you can advance to the second menu by selecting f1.

-ADD TO FILE-

To add to an existing file, select A. Prompts are the same as "New File," except that you exit by selecting N when you are asked if you want to "Add More (Y/N)?".

Adding to a file is a little slower than creating a new file, and the file is not printed to disk and closed until you either select R, Rename and Save File, or f1. You are then asked to name the new file; make sure the name you give it doesn't already exist on the disk. Tax Records 64 automatically writes over and updates files using the original filename, so be careful not to overwrite a file unless that's what you mean to do. (See lines 650 and 7020 in the program: OPEN 1,8,2,"@0:" + N$ + ",S,W". This enables you to save a file using the old name.)

-DELETE FROM FILE-

Select D to delete from the file in memory. You will then be asked if the file is an "I," income, or a "D," deduction.

Next, you'll be asked for at least the first four letters of the file's source. Press the return key and the program will display the first file matching these two descriptions. You'll be asked if this is correct. If it isn't, enter N. The displayed line will be erased and replaced with the next match. This continues until all matches are checked.

When the correct record is displayed, enter Y, and the record will be deleted from the file.

Note: This option works best on small files. As the file gets larger, more and more time is required to shift all the data after each delete. (After I made several deletes to a large file, the cursor seemed to be gone forever.)

A quicker way to delete several records from a sizable file is to change the record (see "Change Data" below). Simply replace the source with the word "delete" (or a similar

word) and enter ".00" for the amount. A little messy, yes, but much faster if you are working with a large file.

If the record to delete is not found in the file, the program will print RECORD NOT FOUND and return to the first menu.

-CHANGE DATA-

Select C to change some part of an existing file. The prompts and displays are similar to Delete and Add. After making any changes, always resave the file. Before you can get to the Work with File menu, where the Quit option is located, your file must be saved.

There is one exception. If you change the file and then select Delete without deleting anything, the program will activate a flag, signaling no change was made, and you could mistakenly advance to Work with File without first having saved the changed file. Always resave your file before turning off your computer. (That is why the Rename and Save File option is included.)

-MERGE TWO FILES-

The program offers the option to merge two files as well as scratch an unwanted file. To merge two files, simply select M and follow the prompts. For instance, you could merge your January and February tax records with this option. Be careful not to merge a file twice, thus duplicating data. You can avoid the Merge option by loading an existing file and adding information to it. However, I find it easier and faster to create monthly files and then merge.

-STATUS CHECK-

Select S to check the status of the disk drive. The disk is automatically checked several times within the program.

-INITIALIZE DISK-

Each time you change a disk, select I to initialize that disk. This is another safety feature. Should you switch disks without initializing, and both the old and new disks contain the same ID code, the disk drive would not realize you had switched disks and could overwrite valuable files or programs.

-MEMORY AVAILABLE-

Select "?" to display how much memory remains unused in the 64. A check may take a minute or more after a sizable file is in memory.

FILE: TAX MERGE

SEARCH:1 1 = CATEGORY 2 = SUBCAT 3 = SOURCE 4 = MONTH 5 = RECORD 6 = AMOUNT

C	SUB	SOURCE	DATE	RECORD	AMOUNT
I	W	SOUTH PAC	04-04	33	$ 1.24
I	F	CALF SALE	01-28	2	$ 154.44
I	D	UT STOCK	04-04	3	$ 32.19
I	I	1ST UNION	04-12	4	$ 10.27
I	R	32 PACK ST	04-01	5	$ 600.00
I	B	MAG SALE	04-02	6	$ 100.00
I	O	TX REFUND	04-15	7	

TOTAL INCOME $ 900.34
TOTAL DEDUCTIONS $ 0

Table 1. *Example of Tax Records 64's printer output according to specified Search (category) for income.*

FILE: TAX MERGE

SEARCH:1 1 = CATEGORY 2 = SUBCAT 3 = SOURCE 4 = MONTH 5 = RECORD 6 = AMOUNT

C	SUB	SOURCE	DATE	RECORD	AMOUNT
D	I	UT STOCK	01-01	1	$ 10.00
D	M	SMITH MD	02-01	2	$ 20.00
D	E	ETSU BOOKS	03-01	3	$ 45.89
D	T	COUNTY	01-04	4	$ 675.19
D	C	BAPT CH	01-18	5	$ 10.00
D	R	IRA KATHY	04-14	6	$ 500.00
D	B	COMPUTER	01-02	7	$ 249.27
D	O	DAY CARE	01-02	8	$ 35.00
D	T	PHONE TAX	01-04	9	$ .94

TOTAL INCOME $ 0
TOTAL DEDUCTIONS $ 1546.29

Table 2. *Example of Tax Records 64's printer output according to Search (category) for deductions.*

FILE: TAX MERGE

SEARCH:3 1 = CATEGORY 2 = SUBCAT 3 = SOURCE 4 = MONTH 5 = RECORD 6 = AMOUNT

C	SUB	SOURCE	DATE	RECORD	AMOUNT
D	I	UT STOCK	01-01	1	$ 10.00

TOTAL INCOME $ 0
TOTAL DEDUCTIONS $ 10

Table 3. *Example of Tax Records 64's printer output according to Search (source) for deductions.*

-HELP !-

If you can't remember what categories or subcategories are allowable, press H for a display of that information.

Let me mention something else here, too. When I first started the program, I intended to disable the run/stop key so you would not accidentally break out (see line 2).

After using this program, however, I found that if you load the C-64 wedge from the demo disk that came with your disk drive before loading Tax Records 64, you can display the disk directory without disturbing the program. This is handy when calling up a file or merging files.

You can break into the program by pressing the run/stop key while at either menu. Print "@$" in Direct mode. This will display the directory. After you have seen all you need, type CONT and press the return key. Now press H twice, and you will return to the menu.

Be careful not to cause a syntax error while in Direct mode.

-WORK WITH AND PRINT FILE-

Select f1 to advance to menu 2, which allows you to examine and manipulate data, print out information contained in the file in memory and quit the program. If you have made changes to the file in memory, remember that Tax Records 64 requires you first to resave the file before you can advance to this menu.

-F2 SCRATCH A FILE-

Select f2 to scratch a file. Be careful with this one. You don't want to erase an important file. To avoid trouble, this option will double-check before it will scratch a file. (Another good reason to use the wedge is to see your disk's directory at a crucial point like this.)

Menu 2's Options

To access menu 2, remember to press f1.

-TOTAL REVIEW-

Select T for a total review of the file. You will be asked if you want a printout. If you don't, you need only press the return key, as this option always defaults to No. The file's records are all displayed on the screen by subcategory.

-SEARCH FOR (BY #)-

You can search for information by Category, Subcategory, Source, Month, Record or Amount.

If you choose one of these options, you will be asked if you want a printout. If you don't, just press the return key. You will then be asked if the search is for an I, income, or a D, deduction. Next, you will be asked to enter information to enable a specific search. For example: Search for all income in the month of 04 (April) or all the deductions under C (contributions).

The screen will display 13 lines of information, then display a Return for More prompt. Along with the information in the file, the program keeps a running total of incomes and deductions, and the total is displayed after the search ends.

The remainder of the Work with File menu contains the same options as the first menu, except Q. This option closes all files and ends the program in an orderly fashion. Remember: If you've made changes in the file, don't quit without first saving the file.

Making It Work for You

This program provides a very helpful tax utility program that should aid you as long as there are taxes—and that will probably be forever.

Of course, you can also use Tax Records 64 at times other than April 15. For instance, you might want to resubmit someone's medical bills for insurance coverage, or to locate important financial information stored in your tax record file, such as interest payments, child care credits and so on.

I've used Tax Records 64 for two years and have given it out to several friends to test and use. Thus, it is a proven product. I think you'll like it, too.

This program occupies a little over 9K of memory before it is run. After it's run, the DIM statements expand this to about twice that, leaving around 20K for your file. An average file takes up 30 bytes, which, with a 500-record file, still leaves 5K unused. Therefore, if you need a file of 600–650 records, you could safely increase the DIM statement in line 340.

One more note: If you accidentally move the cursor while you're typing, reposition it using the cursor keys, type your information and press the return key twice. The second return will accept your input. If you input an incorrect answer,

continue the record until the program asks "CORRECT (Y/N)"; then answer N and press the return key until it returns you to the incorrect line of entry. Retype it and press the return key until you're asked CORRECT (Y/N) again. Answer Y.

Tape users should be able to change lines 650 and 7020 to reflect correct syntax for tape use. Also, delete the OPEN 15 statement in line 6 and the CLOSE 15 in line 1504. R

Address all author correspondence to Gary V. Fields, 86 Lanvale Ave., Asheville, NC 28806.

Scrambler

BY CHUCK McGAFFIN

How good are you at unscrambling words or phrases? In this word game, you must decipher a computer-scrambled word or phrase entered by your opponent. Points are awarded based on the length of the word or phrase and the number of tries taken to correctly unscramble it.

Run it RIGHT

C-64 and VIC-20
Disk or cassette

Scrambler starts with a player typing in a word or phrase that cannot be longer than 30 characters on the Commodore 64 and 15 characters on the VIC-20.

If the input word or phrase cannot be scrambled (for example, "A" or "XXX"), the program will discard the entry and request a new one.

The computer scrambles not only letters, but also numbers, symbols and spaces, then checks to be sure that the scrambled result is different from the original word or phrase. Next, it displays the scrambled result and prompts the player for input of the first character guessed.

As each correct letter is chosen, the completed portion of the unscrambled word is displayed, and the scrambled word is updated to show only those characters remaining. You are then prompted to enter the next character. Wrong guesses are indicated with a musical reprimand.

When the word or phrase has been successfully unscrambled, the player's score is computed according to the length of the word or phrase and the number of errors made during play. The score is then displayed, and Scrambler waits for the player to hit the return key. This pause allows players to inspect the correct word or phrase and the resulting word score before proceeding to the next puzzler.

The VIC-20 version of Scrambler is limited to the input of one word or phrase at a time, while the Commodore 64 version allows each of several players to input a set of words or phrases at the beginning of play.

The running score of each player is displayed along with the score obtained for the current word. The computer prints a prompt to identify which player is to make an entry or to unscramble the currently displayed word or phrase.

Scrambler is a simple yet enjoyable game for young and old. The game also can be used in conjunction with vocabulary lists as a fun way to help children learn to spell.

To conserve memory, the VIC-20 version of Scrambler is limited to one word or phrase at a time. If you have memory expansion, you can easily modify the Commodore 64 version to work on the VIC-20. Aside from the obvious differences between the 40-column and 22-column screens, the only other differences are the initial Pokes to set screen, border and character colors and the sound subroutines (lines 320–360 in the 64 version and lines 245–280 in the VIC version). R

Address all author correspondence to Chuck McGaffin, 21 Maple Ridge, Ballston Lake, NY 12019.

Fly the Grand Canyon

BY THOMAS H. SIMMONDS, JR.

Fly Grand Canyon is an arcade-type, joystick-operated game for the unexpanded VIC-20. You must take off from the airfield and fly through the canyon without hitting walls or planes. If you accrue 2460 points while doing so, YOU MADE IT appears on the screen. Four levels of difficulty are available. They range from a short (S) and possible (P) to a long (L) and impossible (I) game. No one I know of has completed the most difficult level, as the game is programmed.

VIC-20
One joystick required
Disk only

Program Description

The program is written in Basic, with careful attention to the structure of the main game loop to make it as fast as possible. Because of the limited available memory of the unexpanded VIC-20, the program is loaded in two parts:

The Canyon program presents the instructions and Pokes into memory the data for the 60 custom characters. As a final step, the 512 bytes of custom-character memory are protected, and Canyonmod, the main program, is loaded and run by using Pokes to the keyboard buffer.

The second part, Canyonmod, consists of three main sections: initialization, the airport and the canyon maneuvers. The airport portion of the game, which is in a subroutine located at the end of the program, consists of initialization, airport Print statements and plane maneuvers. The next part

contains canyon initialization, the fly-the-canyon loop, the crash and explosion and the score-keeping routines.

Initial Setup

The necessary initial parameters are set up in three areas of the program: at the beginning, to establish the level of difficulty and the screen color, and to activate the custom characters; in the airport subroutine, to establish the Print statements that use the custom characters to create the airport, then to determine the sound, the joystick constants and the initial position of the plane; in the canyon routine, to set up the array dimensions, Print strings and initial position of the canyon opening and plane.

In the Airport subroutine, the plane is Poked to the screen, the joystick direction is detected, and the plane is moved accordingly. If. . .Then statements are included to determine whether the plane is within the screen boundary, to detect whether it has run into any forbidden objects in the airport and, finally, to see whether the plane is headed south to the canyon. When this latter condition is true, the control of the plane is passed to the canyon-flying part of the program. The canyon loop is programmed separately to maximize its speed, as you will see presently.

The Canyon

Initially, a number of housekeeping details are taken care of, including establishing the strings (lines 100–140) that are used to create the random Print statements determining the direction the canyon will turn. Note that, unlike other similar graphics programs that use Print statements of this type, the four E$ Print strings are created using matching custom characters at the edges of the canyon.

The E$ strings are chosen using the E(I,J) array, which allows the program to give the canyon smooth sides as it changes to right, left or straight sections. Line 150 uses two random statements—Y, to locate the initial opening in the canyon, and X, to position the plane. The Poke statement in line 160 locates the cursor one

row up from the bottom of the screen. This is the position of the first canyon Print statement.

Lines 170–270 are the heart of the game. These form the loop that controls the plane, prints the canyon and detects collisions. A number of steps have been taken in these lines to speed up the Basic program. For example, a For. . .Next loop has been used in lines 170 and 270; this is faster than using GOTO 170 in line 270.

The command RND(0) has been used instead of the normal RND(1), and periods (.) have been used to replace the zeroes (0) in lines 170, 180, 190, 210, 240, 250 and 260; again, all this makes Basic run faster. (Many of these speedup ideas came from the excellent article, "Basic Speedup," by John Tanzini, in the March 1984 issue of *RUN*.)

The If statements in lines 170 and 180 randomly choose whether the canyon turns right, left or goes straight. Lines 190 and 200 are If statements that keep the canyon on the screen. In line 210, the plane is advanced down the screen and checked to determine whether it made it through the canyon. For a longer or shorter game, the variable W may be changed in lines 20 and 30.

In line 220, the first statement calculates Q$, the Print string; then the current color of the plane is Poked to the background color in preparation to moving it. The Q$ is printed at the bottom of the screen and all other canyon Print statements are pushed up, giving the illusion that the plane is moving down the canyon. Next, the position of the plane is updated and the color Poked to yellow to make it visible again. Finally, K is given the value of X. A collision with the wall or another plane is detected with the If. . .Then statement in the next line.

Line 240 Pokes in randomly colored planes. The frequency of their occurrence is controlled by the variable D, which was set by the possible/impossible option. You can make the game easier or more difficult by changing the value of the variable D in line 20.

The next two lines read the joystick, change X, the position of the plane, and set P, the custom character for a right, left or straight airplane. The last line of the loop is a Next statement and sends the program through the loop again.

Final Routines and Future Fun

Lines 280 and 290 produce the visual and sound effects of the crash. The screen is shaken by Poking the address that locates the center of the screen at the same time the screen colors are randomly changed. Following the explosion, the game score is updated and comments on the results printed to the screen. The player is then asked to hit the joystick's fire-button to play another game.

The program is not long and lends itself to modification. The canyon Print strings in lines 100–140 are composed mostly of randomly chosen graphics characters. As you play the game, watch what happens to these Print statements. Note that the individual characters change. This is the result of the graphics characters being taken from a part of memory that is the Basic program rather than the character ROM.

The canyon opening and the immediate characters on each side are critical, however. You might try changing the width of the canyon to make the game more difficult or easy. By modifying line 240, some other obstacle besides an airplane could be introduced to the canyon.

If the game is too hard, you may set the RND statement to some value other than >0.4. Go higher to >0.67, and more straight section of canyon will be chosen, making it easier to win. Experiment and have fun! R

Address all author correspondence to Thomas H. Simmonds, Jr., 127 Chestnut St., North Andover, MA 01845.

Play Me A Color

BY JOSEPH T. WOYTON

Explore the mysteries of machine language programming with this tutorial, which describes how to make your VIC into a color organ.

Basic programs often require the use of lookup tables to compare a series of data for coincidence, equality or tests of validity. This is usually done with For. . . Next loops, and the computer may take several seconds execute a lengthy comparison list in Basic. This can certainly slow down your program's action.

VIC-20
Disk only

Using a machine language program, as illustrated here, instead of the For. . . Next loop, results in much faster computer processing. A machine language program, with its blazing speed and efficiency, will run hundreds of times faster than its Basic counterpart. When you press a key, you want action!

About Machine Language

For the machine language novice who has no assembler or monitor programs, the following description will show that it is fairly easy to implement simple machine language programs by using only VIC Basic.

Machine language uses only numbers for machine instructions. All information must be in the form of integer decimal values from 0 to 255 for entry via Basic. Memory addresses are identified by page (high address byte), with 256 locations (low address byte) per page, to format the two-byte machine language address the computer requires.

The machine operation codes are also specified in decimal values. These code numbers control the computer functions as the computer sequentially steps through the machine language program. (See Table 1 for procedures to calculate memory addresses and operation codes for machine language programs.)

As in assembly language, using mnemonics for reference purposes helps us bridge the gap between our English language and the numbers-only world of machine language. Mnemonics are programmers' English abbreviations for the operations specified by the numeric operation codes. (See Table 2 for a complete explana-

---

The VIC memory addresses are specified as quantities from 0 to 65,535. Convert these numbers to the two-byte format by following these examples.

ADDRESS 197, the VIC keyboard:

PAGE = INTEGER (ADDRESS/256)	LOCATION = ADDRESS – PAGE*256
PAGE = INTEGER (197/256)	LOCATION = 197 – 0*256
PAGE = 0 (high address byte)	LOCATION = 197 (low address byte)

ADDRESS 828, the start of the cassette buffer:

PAGE = INTEGER (828/256)	LOCATION = 828 – 3*256
PAGE = 3	LOCATION = 60

ADDRESS 36879, the VIC color register:

PAGE = INTEGER (36879/256)	LOCATION = 36879 – 144*256
PAGE = 144	LOCATION = 15

The VIC operation codes are usually specified as hexadecimal (HEX) quantities. Convert these to decimal values by following these examples.

HEX 0 to 9 = DECIMAL 0 to 9
HEX A, B, C, D, E, F = DECIMAL 10, 11, 12, 13, 14, 15
Load the accumulator = OP CODE HEX A9
HEX A9 = 10*16 + 9 = 169 DECIMAL
Store the accumulator = OP CODE HEX 8D
HEX 8D = 8*16 + 13 = 141 DECIMAL
Branch if result zero = OP CODE HEX F0
HEX F0 = 15*16 + 0 = 240 DECIMAL

Table 1. *Memory address and operation code conversion.*

tion of all operation codes used in this program.)

The machine language program is placed into a safe and convenient memory location (it won't interfere with the Basic program) using Read, Poke and Data statements in Basic. The cassette buffer memory area, addresses 828–1019, is an excellent storage place. The Color Organ program uses addresses 828 to 948.

The *VIC-20 Programmer's Reference Guide* contains more on machine language programming. You should read this or other reference material to become more familiar with the VIC's 6502 microprocessor functions.

The Machine Language Program

In going through this description, you'll note many refer-

Op Code	Mnemonic	Operation
169	LDA#	Load the accumulator with the number in the next byte.
141	STA	Store the accumulator contents in the memory address given by the next two bytes.
173	LDA	Load the accumulator with the contents of the memory address given by the next two bytes.
201	CMP#	Compare the contents of the accumulator with the number given in the next byte.
240	BEQ	Branch forward or backward by the number of steps given in the next byte, *if* the result of the previous operation was zero (equality). Backward = 256 – steps.
32	JSR	Jump to the subroutine at the address given by the next two bytes. Save the current return address.
76	JMP	Jump to the address given in the next two bytes.
162	LDX#	Load the X-index register with the number given in the next byte.
221	CMP(X)	Compare the contents of the accumulator with the number at the address given by the next two bytes plus the value in the X-index register.
202	DEX	Decrement the value in the X-index register by one.
48	BMI	Branch by the number of steps given in the next byte, *if* the result of the previous operation was negative.
96	RTS	Return from this subroutine.
189	LDA(X)	Load the accumulator with the number at the address given by the next two bytes plus the value in the X-index register.

Table 2. *Operation codes, mnemonics and operations.*

ences to accumulator operations. The accumulator is the main processing register of the microprocessor. It is used to transfer data, make comparisons and perform arithmetic operations.

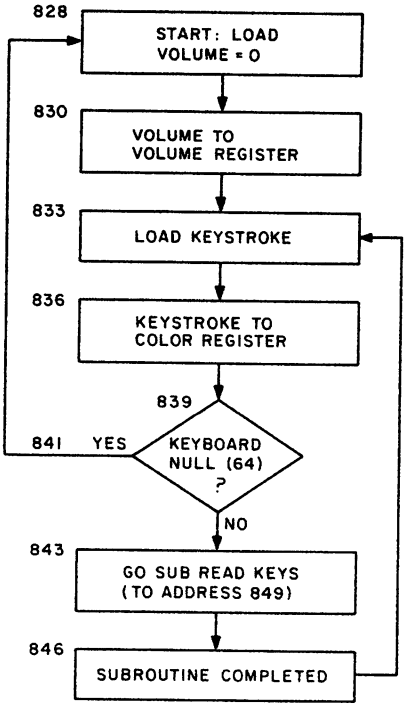
Compare the program flowchart and machine language listing as we discuss the major steps in the lookup-table routine. The listing has liberal comments to help explain the machine language program activities.

Starting at address 828 (p. 3, location 60) the machine language program sequentially executes each operation code. The keyboard entry obtained from address 197 is loaded (LDA) into the accumulator. This value is stored (STA) into the VIC color register to create a screen and border color. If you press any key, the program goes to JSR, the Read Keys subroutine.

The subroutine compares (CMP(X)) the accumulator to each of the values in the Key Data table. This uses an index address technique, where the microprocessor's X-index register is used as a pointer that steps down (DEX) the data table.

When a match is found (BEQ), the accumulator loads (LDA(X)) the proper value from the Tone Data table by using the X-index pointer value as a reference. The tone value is then stored (STA) into the VIC sound register to produce an organ tone.

FLOWCHART: MAIN PROGRAM



NOTE:
NUMBERS REFERENCE MEMORY ADDRESSES

Fig 1. *Flowchart of Color Organ program.*

If no key match is found, the X index will be decremented (DEX) below zero. The program branches on this negative (BMI) to return (RTS) from the Read Keys subroutine. It then goes to look (JMP) for another key press at the top of the ML program.

The Key Data and Tone Data are thus used in pairs, starting at the end of the tables (X index = 36) and working backwards (DEX) to the beginning (X index = 0). For example, the keyboard £ (code 6) in address 911 is used with the musical tone C4 (code 240) in address 948.

This offset relationship holds, stepping through both data tables. You can easily change the keyboard tone as-

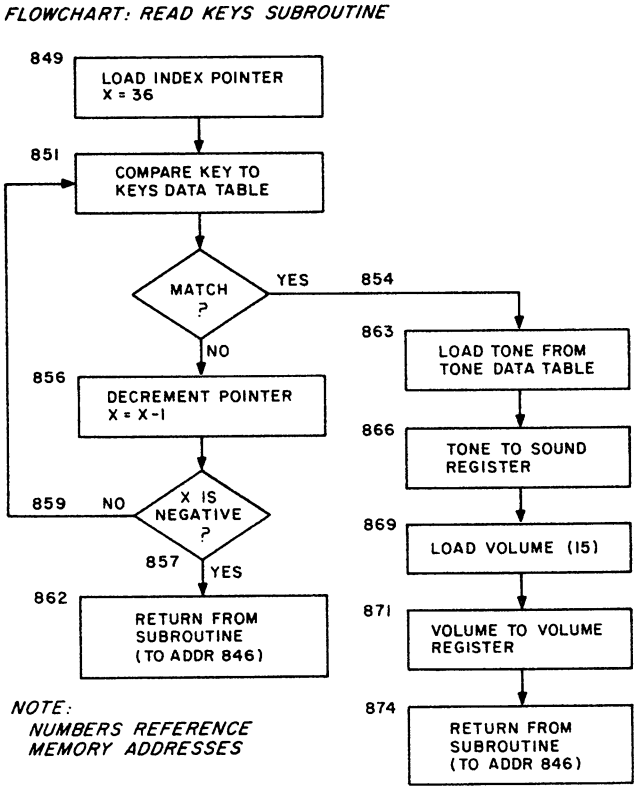


Fig. 2. *Flowchart of Read Keys subroutine.*

signments by rearranging the data in either data table. You can make the key pattern more like the standard piano layout (white and black) if you wish. You may also substitute your own data to construct any other kind of lookup table.

Note that the data in address 850 identifies the number of values in each of the tables. This program uses 37 entries, 0–36. Insert the proper value (up to 255) for your own data list. Changes to the machine language program are made by modifying the values in the Basic program's Data statements.

The Basic Program

In the Basic program's operation, the machine language routine is Read and Poked into memory (line 100), starting at address 828. The machine language program is entirely contained within the Data statements (lines 101–106). The screen is cleared and prompt messages are displayed (lines 110–140).

The machine language program is then called from Basic as a subroutine by SYS 828 (line 150) to play the VIC Color Organ.

Good luck and have fun. If this is your first attempt at machine language programming, you are about to enter a new dimension in computer power and speed. R

Address all author correspondence to Joseph T. Woyton, 106 Braddock Drive, Mauldin, SC 29662.

Table 3. *Machine language listing.*

Memory Address	Page 3 Location	OP Code	Mnemonic	Comments
828	60	169	LDA#	START: load volume
829	61	0	—	Volume = 0
830	62	141	STA	Poke volume to
831	63	11	—	low address (location)
832	64	144	—	high address (page)
833	65	173	LDA	LOAD KEYSTROKE from keyboard
834	66	197	—	low address

Memory Address	Page 3 Location	OP Code	Mnemonic	Comments
835	67	0	—	high address
836	68	141	STA	Poke color to
837	69	15	—	low address
838	70	144	—	high address
839	71	201	CMP#	Check for no key pressed
840	72	64	—	64 is keyboard null
841	73	240	BEQ	To START if no key,
842	74	241	—	branch back 15 steps
843	75	32	JSR	To READ KEYS subroutine
844	76	81	—	low address
845	77	3	—	high address
846	78	76	JMP	To LOAD KEYSTROKE
847	79	65	—	low address
848	80	3	—	high address
849	81	162	LDX#	READ KEYS
850	82	36	—	37 data points
851	83	221	CMP(X)	COMPARE to KEY DATA table
852	84	107	—	low address
853	85	3	—	high address
854	86	240	BEQ	To PLAY if match,
855	87	7	—	branch forward 7 steps
856	88	202	DEX	Next key data
857	89	48	BMI	If end of key data, branch
858	90	3	—	forward 3 steps to RTS
859	91	76	JMP	To COMPARE, repeat
860	92	83	—	low address
861	93	3	—	high address
862	94	96	RTS	RETURN from subroutine
863	95	189	LDA(X)	PLAY: load tone data from
864	96	144	—	low address

Memory Address	Page 3 Location	OP Code	Mnemonic	Comments
865	97	3	—	high address
866	98	141	STA	Poke tone to
867	99	11	—	low address
868	100	144	—	high address
869	101	169	LDA#	Load volume
870	102	15	—	15 is max volume
871	103	141	STA	Poke volume to
872	104	14	—	low address
873	105	144	—	high address
874	106	96	RTS	RETURN from subroutine
875	107	17	KEY DATA	A
876	108	41		S
877	109	18		D
878	110	42		F
879	111	19		G
880	112	43		H
881	113	20		J
882	114	44		K
883	115	21		L
884	116	45		:
885	117	22		;
886	118	46		=
887	119	48		Q
888	120	9		W
889	121	49		E
890	122	10		R
891	123	50		T
892	124	11		Y
893	125	51		U
894	126	12		I
895	127	52		O
896	128	13		P
897	129	53		@
898	130	14		*

Memory Address	Page 3 Location	OP Code	Mnemonic	Comments
899	131	0		1
900	132	56		2
901	133	1		3
902	134	57		4
903	135	2		5
904	136	58		6
905	137	3		7
906	138	59		8
907	139	4		9
908	140	60		0
909	141	5		+
910	142	61		–
911	143	6		£
912	144	135	TONE DATA	C(1)
913	145	143		C#
914	146	147		D
915	147	151		D#
916	148	159		E
917	149	163		F
918	150	167		F#
919	151	175		G
920	152	179		G#
921	153	183		A
922	154	187		A#
923	155	191		B
924	156	195		C(2)
925	157	199		C#
926	158	201		D
927	159	203		D#
928	160	207		E
929	161	209		F
930	162	212		F#
931	163	215		G
932	164	217		G#
933	165	219		A
934	166	221		A#

Memory Address	Page 3 Location	OP Code	Mnemonic	Comments
935	167	223		B
936	168	225		C(3)
937	169	227		C#
938	170	228		D
939	171	229		D#
940	172	231		E
941	173	232		F
942	174	233		F#
943	175	235		G
944	176	236		G#
945	177	237		A
946	178	238		A#
947	179	239		B
948	180	240		C(4)

Bombs Away

BY JAMES F. McCONNELL

Tank Defense is a colorful arcade-style game that requires quick reflexes and strategy. The screen displays a continuous rain of lethal bombs falling from the sky. On the ground is a typical community, complete with buildings, a highway with traffic—and a tank.

Your only defense against the incessantly falling bombs is to shoot them out of the sky. Fortunately, you are an expert marksman and have at your disposal a state-of-the-art weapons system in your tank. You move the tank back and forth across the screen to adjust the trajectory and can select either of two cannons to fire. The joystick controls these options as follows: L—move tank left, R—move tank right, Up—fire left cannon, Down—fire right cannon.

VIC-20
One joystick required
Disk or cassette

You gain five points for exploding a bomb before it hits the ground and lose one point for every time you shoot and miss. The game will end abruptly if a bomb hits a building, a truck or the tank. It will also end if you accidentally shoot one of the trucks or buildings. Each time a bomb hits the ground without causing damage, the falling bombs will be fewer and faster, and the game will end if this happens five times.

The game starts out slowly enough so that you shouldn't have any difficulty intercepting and exploding the falling bombs. However, the game's pace gradually increases, and after about a minute and a half of playing time, you'll be hard-pressed to keep up with the action.

The Program

The program is written as

two programs. The first provides instructions for playing and loads the game's custom character set into the VIC's memory. The second program, which contains the main game, then loads automatically.

The star of this program is the tank, constructed from six independently customized characters. The instructional value of this game lies in the method used to provide smooth, coordinated and fairly rapid animation of so many characters at once. If the tank were animated with Peeks and Pokes, the result would be herky-jerky, disjointed and slow. The tank in this program is animated with Print statements using the MID$ function. Line 21 of the program accomplishes the whole thing. Study this line to understand how it works.

The SYS 7432 in line 10 causes the VIC to jump to a short machine language subroutine for reading the joystick. This routine is Poked into a portion of the custom-character memory along with the custom characters in the first program. The memory used stretches from location 7432 to 7496, a section of memory that normally contains the information for characters CHR$(33)–CHR$(41). Since these characters are not used in the program, this area of memory is a relatively safe place to store this subroutine.

Those who like to tinker and modify might try altering the colors I chose for the program. I tried dozens of color combinations before I finally made up my mind.

The pace of the game is controlled in the main loop between line 21, which moves the tank, and line 30, which redirects to the start of the loop. Anyone who thinks the game is too easy can significantly increase the game's pace by removing these lines and changing line 30 to GOTO 200. Although I feel that the playability of the game suffers when this is done, it does serve to illustrate the smoothness and rapidity of this method of animation. R

Address all author correspondence to James F. McConnell, Box 111, RD #1, Marathon, NY 13803.

Trapped in the Maze

BY JAMES MILLER

This article takes you from keyboard character movement through random maze construction, to a complete program, using color, sound and custom characters.

When I first tried simple game programming on the VIC, I was stumped! Poring over scores of game programs and tutorials didn't help much, either. I thought that it was just too tough!

VIC-20
Disk or cassette

I believed there was some secret (and very complex) formula that controlled everything, a formula that the great gamers knew by intuition and were not about to share with slow-witted folks like me!

But that was my problem. . .I was looking for a solution far more difficult than it really turned out to be! The simplicity of Basic game design is so apparent that I missed it completely!

Maybe that's been your problem, too? If this is the case, let's run through some simple ideas and concepts that really control game action and see how it's done.

This tutorial is for those of you who are too often mystified by computing and want some elementary, step-by-step help!

Screen and Color + Get

After many attempts at Poking and printing graphics and trying, unsuccessfully, to get them to move up and down and right and left on

the screen, I finally hit upon a solution.

Poking movement in one direction is easy, but adding the Get command finally gave me four-way keyboard control. The core routine I devised for character movement was so direct and simple that I almost doubted its ability.

If you've dabbled with Pokes at all, you've learned that the computer screen is divided into locations for character and color. On the VIC, the screen begins at location 7680 (upper-left corner) for character placement. For color values, the screen begins at 38400.

From the top left, there are 506 possible Poke locations within a 23-row by 22-column screen.

You can use any part of this total working area. Here's some simple math to keep in mind. If x = your starting point, then:

x + 1 = one space right
x − 1 = one space left
x + 22 = one row down
x − 22 = one row up

Using a GET A$ command combined with a blind Goto loop is just about all you need to create and control a graphics character's movement on the screen.

I'll look at the core routine first. Next, I'll develop and expand it, adding sound and color, then custom characters and a maze. Finally, I'll turn it into a strategy game you play against the clock.

The Core Routine

The core program in Listing 1 is only 12 lines long and could be shortened if you used a lot of multiple statements. (Note: Listings 1 and 2 are not included on the ReRUN tape or disk.) The routine takes care of four basic action elements: choosing a starting location for the character, setting a color-value constant, Poking the character and its color to the screen and moving it around.

Line 10 clears the screen.

Line 15 initializes two variables. X is the first screen character location, and CL is the color constant used to color the character.

Line 22 Pokes a ball character (81) in the first screen position and colors it black.

Line 30 sets up the Get routine.

Line 32 just provides a brief delay loop to slow things down a bit.

Line 33 Pokes location 650 so that all keys will repeat as you hold them down. We're only concerned with four keys for movement in this routine, however.

Lines 35-50 set up four If...Then conditional statements that provide for character movement depending on which key you press—the up arrow, left arrow, R and D keys move the ball as you would expect: up, left, right and down, respectively.

Line 60 loops back to line 20 continuously. This ensures that all Pokes and key presses are read instantly, to keep movement going at a brisk pace.

The routine works alone, as it is, to demonstrate how movement is controlled. We're not even erasing the ball as it moves around the screen—that comes later!

You can alter the keys chosen for movement to any others you think are more convenient, as long as you remember what they are! In the final game program, we'll use the cursor control keys plus f5 and f7 for left, right, up and down, because they're close together and much easier to manipulate quickly in game action.

Return now to line 15 and change X to another screen value larger than 7680 but less than 8185 (the last location on the screen). Try 7910, for example. As you run the program again, the ball appears at screen center, but you still have complete control of its movements.

Some games begin with the primary character at the bottom of the screen, and the basic moves are towards the top of the screen. In that case, there's no need for a Down command—like line 50. In fact, it's a good idea to exclude it in some cases, or you'll find your character dropping off the bottom of the screen, into never-never land!

Building a Game

From the core program, you can expand the simple Get movement technique to include routines that begin to create a game atmosphere. The program in Listing 2 is just twice as long

as the core routine, and if you study the listing, you'll see it's all there, with only slight changes.

Lines 16-20 are new and initialize new parts of the game structure we're planning. BR in line 16 sets up 125 barrier elements (something to avoid or run into as you move the ball about the screen!). Y is the starting location for the barriers (7702 is one line down from top left on the screen).

Look at lines 17-18. They serve as a loop that does no more than Poke 125 random locations on the screen with our barrier character, {CMD + }, and colors each one blue.

When you ran the core routine, the ball character appeared first, but in this program, all 125 of the barriers pop into place before the ball appears.

Line 19 takes care of three functions necessary in the development of the game. First, it Pokes a goal location (8185) and places a purple symbol that looks somewhat like a # there. That's where we're headed with the ball!

Second, the VIC's clock function, TI$, begins a counting routine that is reset to 0 every time the program encounters line 19.

Finally, location 198 is Poked with a 0 so that the keyboard buffer won't store any keystrokes (a value of 10 is normal). As you race across the screen, extra keystrokes can be stored in a buffer, thus causing extra character movement when you don't want it.

Line 20 represents a change from the core routine. The REM has been deleted and replaced by a random variable that changes the color of the ball. You remember that line 60 was a Goto statement that sent the program back to line 20.

This adds a nice touch to the program and also helps you know exactly where you are. The ball changes color several times a second and you can turn back on yourself without worry because the color change tells you your position! That would not be the case if we used a single, unchanging color Poke.

Line 22 Pokes the ball character *after* the new color variable; also, the

color Poke has been changed to include variable CX, the random color value. Line 29 also is new and prints the clock at the top right of the screen. If more than 29 seconds tick off, the game branches to a new subroutine.

Lines 30-60 remain just like the core routine, but notice the addition of lines 52 and 54. Those pair of IFs are statements that provide the only way out of the Goto loop. In other words, they branch the game to two other routines that add flexibility to the program.

First, if Peek (the # value of) location 8185 is ever 81 (the ball), then the program branches to line 100 to tell you you're free!

If, however, you run into a barrier, the program jumps to line 65 and then to line 200 as a * replaces the {CMD +}, and you get an OUCH! message. That also happens if you run out of time.

Without at least these two branching statements, you'd be locked into the main loop forever, able only to move around, without doing much at all.

Let the Game Begin

You are now familiar with moving a single object around the screen, and perhaps you've even experimented with another character in place of the ball (there are lots of choices).

You've also learned to create barriers, which are at the heart of all maze games. But so far, we really don't have a game at all.

Putting a complete game program together and making it entertaining or exciting takes some added thought. What's the object of the game? What other elements do we add to fill in the blanks?

Sound, Strategy and Custom Characters

By its very nature, a maze game involves escape and, usually, a race against the clock. In addition, most of us like to see some progress or reward, and that's where scores come in.

Beyond that, you may want to explore more of your computer's capabilities, especially its talent for special characters you create in place of common keyboard

graphics.

The main Frenzy program takes care of all of that and a little more in just 58 lines. Even with the addition of custom characters, there's plenty of memory left for you to experiment, modify and change the program until you have just what you want in a maze game!

The core remains the same, and the branches used in Listing 2 are still there—they're just changed to include scoring, sound and color.

In addition, several new elements have been added: custom characters, several sound routines, screen color changes, a way to animate the moving character and a way to end the game.

Lines 2-7 of the main program take care of special character development by changing three standard keyboard characters (@, [,]) to three created characters that add interest to the game.

In place of a ball is a creature that you can move around the screen. The barriers, too, are customized. Also, a new character is used as an added element in the game. . .a bomb to defuse for points.

Look at lines 35-50. You will see one new command in each line. As the creature moves about the screen, these commands act as an eraser; spaces behind it are Poked with a blank so that the creature doesn't leave a trail as he moves.

Lines 35-50 also change the movement keys, to make things easier on our fingers.

Pressing the cursor-down key moves the creature left only, the cursor-right key moves him right, f5 moves him up and f7 moves him down.

Escapes to the goal (#) earn points, and defusing the bombs earns a bonus. The bombs are randomly Poked on the screen, and there's plenty of time to get most of them.

As an additional element, the maze grows more complex as you advance from round to round, but it's worse on you if you slam into a barrier—points are lost and the maze really grows!

Below, you'll find an explanation of this program's important routines, plus a brief description of the variables and suggestions about

changes you may make.

Lines 1-7: Title, lower memory, the creation of custom characters in Data statements.

Lines 8-9: Initialize score and rounds plus barriers.

Lines 10-16: Clear screen, set volume and sound variables plus screen and color locations for each custom character.

Lines 17-22: Poke and randomize character placement, set time TI$ to 0.

Lines 30-60: Get/Goto character movement loop plus conditional branch statements.

Lines 100-110: Routine for incrementing and reporting the score if the goal is reached, plus sound and color changes.

Lines 200-210: Routine for score decrement if barrier hit, plus sound and color changes.

Lines 300-360: Game end routine.

Lines 600-610: Title screen character Poke routine.

SK = score, RO = round
BR = barriers
V = volume
S1, S2 = sounds 1 & 2
X = beginning character location
CL = color constant
Y = beginning location for barriers
Z = beginning location for bombs
B = random # of bombs
CY = random location of bombs
CX = random color for moving creature

Table 1. *Frenzy program variables.*

Remember that the key to this program remains as simple as the core routine: A short Get/Goto loop takes care of all movement, and the addition of a few If. . .Then branching statements takes care of all the rest—scoring, sound and color. R

Address all author correspondence to James Miller, 2142 Odema Drive, Lima, OH 45806.

Passage to Zxylon

BY CHUCK MOUNTS

Zxylon is a game that requires a little logic, skill and luck. The time is 2281 A.D., and your spacecraft has landed in the interior of planet Zxylon. You must retrieve the yellow plutonium (vital to your planet's existence), which was stolen by the Zxylon warriors, and avoid being captured by the deadly Zxyclops that guard it. But be careful—the walls of the caverns are electrified.

VIC-20
One joystick required
Disk or cassette

How to Play

To retrieve the stolen plutonium, you maneuver your Being around the screen with a joystick. Lure the Zxyclops into the electrified walls to destroy them. You have 30 seconds to capture the egg before it hatches, unleashing the invincible green Zxyborg. The only weapons you carry are three hyperspace bombs, which you may release by pressing your joystick's fire-button.

You have five men per game and are awarded an additional man after scoring 1000 points. You gain ten points for each Zxyclops you destroy, 25 points for each plutonium mound you recover and 50 points for capturing the egg before it hatches. Each time you destroy all the Zxyclops guarding the plutonium and capture the egg, an additional Zxyclops is added to guard the plutonium for the next stage.

The program uses random locations (the RND statement)

to set up the screen, so the walls of the caverns, the placement of the plutonium and the placement of the Zxyclops are different for each stage and game. In the upper left-hand corner of the screen is displayed the number of your remaining men. The upper middle of the screen displays the number of hyperspace bombs that remain, and in the upper right-hand corner is displayed your score.

The Program

The program is set up in two parts, making it compatible to run on the standard 5K unexpanded VIC-20. The Zxylon program sets up the programmable character set and instructions, and gives a brief scenario of the game, along with an interesting graphics display.

Zxylonmod is actually the game itself and uses almost all of the VIC's memory. Written entirely in Basic, the game includes high-resolution graphics and sound in addition to multi-colored graphics.

After you've loaded Zxylon into the VIC's memory, type RUN. Leave the play button on the cassette recorder depressed and do not rewind the tape. Zxylon will automatically clear itself out of the VIC's memory and load Zxylonmod.

I believe you'll find Zxylon to be an exciting and almost addicting game to play. R

Address all author correspondence to Chuck Mounts, 1598 Secretariat Drive, Annapolis, MD 21401.

Bug-in-a-Maze

BY JAMES R. MILLER

Using custom characters and a constructed maze, this multi-level addition review for primary grade-schoolers adds entertainment and reward to the practice of early mathematics skills.

The Bug-in-a-Maze program provides review of addition for children in primary grades. It uses custom characters and a constructed maze to entertain and to reward the practice of elementary math skills.

Bug-in-a-Maze includes nine levels linked to a pair of random-number generators and the level factor (L). Selecting level 3, for example, will generate addition problems that are all multiples of 3. The level factor also determines the problems' complexity. Level 1 is the easiest; level 9 the most difficult.

Bug-in-a-Maze is good fun, too. Whenever a right answer is given, a nervous little bug appears at the top opening of the maze and begins to scramble through the twists and turns at a speed that is linked to the value of X (the first number in each addition problem).

As a child tries more difficult problems, the crazy creature will run faster and faster through the maze, changing color with each level. At the bottom, the bug turns black and settles fitfully into his nest, ready to run again. Pressing the return key restarts the game.

If an incorrect answer is given, a second black bug appears out of nowhere, eats the

VIC-20
Disk or cassette

wrong answer and displays the correct answer in the center of the maze. After a brief pause, the screen clears and the game automatically begins again.

The Program

The program employs a fairly common routine to create a user-defined character in place of one of the normal characters found in the standard VIC set (contained in ROM).

Lines 110–140 move the standard set out of ROM into RAM and read a set of data numbers to change the @ symbol to the bug used in the game. There is also an instruction to lower memory a bit to protect the Basic program from the RAM character set.

The maze through which the bug dashes is constructed by means of Print statements rather than random Pokes. This means the bug always runs the same pattern (although at varying speeds).

The creature already skips one opening (on purpose) and then runs back again to drop through to the next level.

Making Changes

While Bug-in-a-Maze performs only addition problems now, it's easy to make changes to include subtraction or multiplication. Simply change the variables in lines 169–185 to (*) or (–) instead of (+).

You also can change the bug character (a character-creator utility program is a great help in this case). If you do attempt to alter the bug to some other design, do so before you run the program: Go to line 130 and change each of the eight numbers to whatever you've selected.

Try the following sets of data to see my alternative creatures.

```
DATA 129,90,36,126,129,219,66,60
DATA 102,153,36,36,24,102,66,231
DATA 24,36,24,102,90,36,36,102
```

Running the program after entering the new data should give you a brand new creature! R

Address all author correspondence to James Miller, 2142 Odema Drive, Lima, OH 45806.

Please send me back issues of ReRUN

____ Cassette version(s) at $11.47* ____ C-64 ____ Vol. 1
____ Disk version(s) at $21.47* ____ VIC-20 ____ Vol. 2

* Prices include $1.50 postage and handling. Foreign Airmail please add an additional $1.50 per item. U.S. funds drawn on U.S. banks only.

☐ Check/MO ☐ MC ☐ VISA ☐ AE

Card # Exp. Date

Signature

Name

Address

City State Zip

ReRUN • 80 Pine Street • Peterborough, NH • 03458

BEAT THE RUSH—
Order the next edition now!

Please send me the ReRUN Summer Edition:

_____ Cassette version(s) at $11.47 ea.*
_____ Disk version(s) at $21.47 ea.*

* Prices include $1.50 postage and handling. Foreign Airmail please add an additional $1.50 per item. U.S. funds drawn on U.S. banks only.

Summer Edition (available in June) runs on C-64 only.

☐ Check/MO ☐ MC ☐ VISA ☐ AE

Card # Exp. Date

Signature

Name

Address

City State Zip

ReRUN • 80 Pine Street • Peterborough, NH • 03458

Your Favorite Programs from RUN December, January and February!

Featuring:
- *A SPREADSHEET
- *GAMES
- *UTILITIES
- . . . and more!

If any manufacturing defect becomes apparent, the defective cassette or disk will be replaced free of charge if returned by prepaid mail within 30 days of purchase. Send it, with a letter specifying the defect, to:

ReRUN • 80 Pine Street • Peterborough, NH 03458

Replacements will not be made if the cassette or disk has been altered, repaired or misused through negligence, or if it shows signs of excessive wear or is damaged by equipment.

The programs in ReRUN are taken directly from listings prepared to accompany articles in *RUN* magazine. They will not run under all system configurations. Use the RUN It Right information included with each article as your guide.

The entire contents are copyrighted 1985 by CW Communications/Peterborough. Unauthorized duplication is a violation of applicable laws.

© Copyright 1985 CW Communications/Peterborough

CW COMMUNICATIONS/PETERBOROUGH